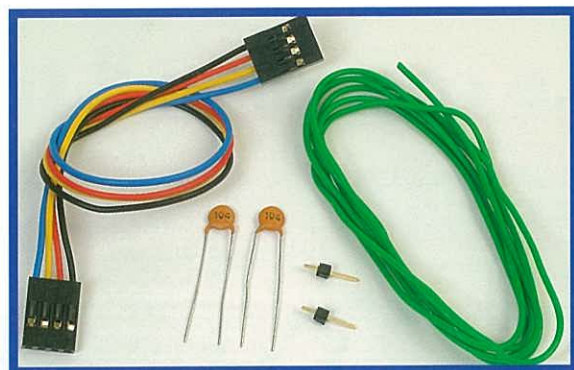
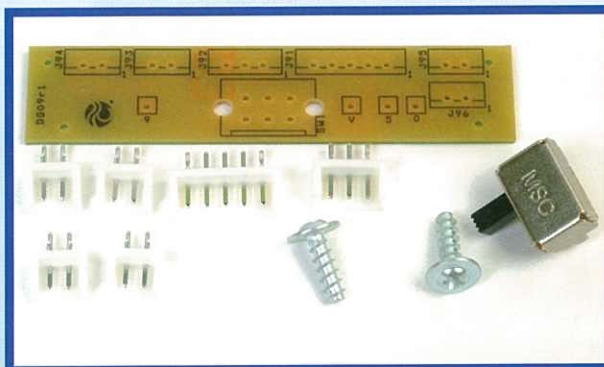
Direttore responsabile:
ALBERTO PERUZZO
Direttore Grandi Opere:
GIORGIO VERCELLINI
Consulenza tecnica e traduzioni:
CONSULCOMP S.n.c.
Pianificazione tecnica
LEONARDO PITTON

Direzione, Redazione, Amministrazione: viale Ercole Marelli 165, Tel. 02/242021, 20099 Sesto San Giovanni (Mi). Pubblicazione settimanale. Registrazione del Tribunale di Monza n. 1738 del 26/05/2004. Spedizione in abbonamento postale gr. II/70; autorizzazione delle Poste di Milano n. 163464 del 13/2/1963. Stampa: Staroffset s.r.l., Cernusco S/N (MI). Distribuzione SO.DI.P. S.p.A., Cinisello Balsamo (MI).

"ELETTRONICA DIGITALE" si compone di 70 fascicoli settimanali da suddividere in 2 raccoglitori.

RICHIESTA DI NUMERI ARRETRATI. Per ulteriori informazioni, telefonare dal lunedì al venerdì ore 9.30-12.30 all'ufficio arretrati tel. 02/242021. Se vi mancano dei fascicoli o dei raccoglitori per completare l'opera, e non li trovate presso il vostro edicolante, potrete riceverli a domicilio rivolgendovi direttamente alla casa editrice. Basterà compilare e spedire un bollettino di conto corrente postale a PERUZZO & C. s.r.l., Ufficio Arretrati, viale Marelli 165, 20099 Sesto San Giovanni (MI). Il nostro numero di c/c postale è 42980201. L'importo da versare sarà pari al prezzo dei fascicoli o dei raccoglitori richiesti, più le spese di spedizione € 3,10 per pacco. Qualora il numero dei fascicoli o dei raccoglitori sia tale da superare il prezzo globale di € 25,82 e non superiore a € 51,65, l'invio avverrà per pacco assicurato e le spese di spedizione ammonteranno a € 6,20. La spesa sarà di € 9,81 da € 51,65 a € 103,29; di € 12,39 da € 103,29 a € 154,94; di € 14,98 da € 154,94 a € 206,58; di € 16,53 da € 206,58 in su. Attenzione: ai fascicoli arretrati, trascorse dodici settimane dalla loro distribuzione in edicola, viene applicato un sovrapprezzo di € 0,52, che andrà pertanto aggiunto all'importo da pagare. Non vengono effettuate spedizioni contrassegno. Gli arretrati di fascicoli e raccoglitori saranno disponibili per un anno dal completamento dell'opera. IMPORTANTE: è assolutamente necessario specificare sul bollettino di c/c postale, nello spazio riservato alla causale del versamento, il titolo dell'opera nonché il numero dei fascicoli e dei raccoglitori che volete ricevere.

IN REGALO in questo fascicolo

1 Cavetto da 4 fili con due connettori volanti femmina a 4 vie
1 Filo verde rigido
2 Connettori diritti, maschio a 1 via
2 Condensatori ceramici da 100 nF

IN REGALO nel prossimo fascicolo

1 Scheda DG09r1
1 Commutatore 2 posizioni 2 vie a 90°
4 Connettori maschio da c.s. diritti a 5 vie
1 Connettore maschio da c.s. diritto a 3 vie
2 Viti 3,1x10 mm

COME RACCOGLIERE E SUDDIVIDERE L'OPERA NELLE 4 SEZIONI

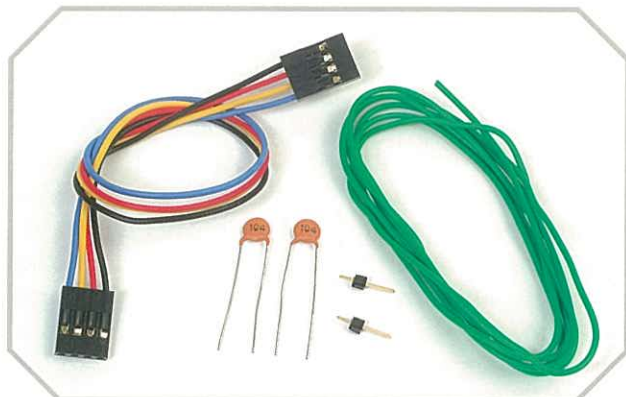
L'Opera è composta da 4 sezioni identificabili dalle fasce colorate, come indicato sotto. Le schede di ciascun fascicolo andranno suddivise nelle sezioni indicate e raccolte nell'apposito raccoglitore, che troverai presto in edicola. Per il momento, ti consigliamo di suddividere le sezioni in altrettante cartellette, in attesa di poterle collocare nel raccoglitore. A prima vista, alcuni numeri di pagina ti potranno sembrare ripetuti o sbagliati. Non è così: ciascuno fa parte di sezioni differenti e rispecchia l'ordine secondo cui raccogliere le schede.
Per eventuali domande di tipo tecnico scrivere al seguente indirizzo e-mail:
elettronicadigitale@microrobots.it

Hardware Montaggio e prove del laboratorio

Digitale di base Esercizi con i circuiti digitali

Digitale avanzato Simulazione con MPLAB (II)

Microcontroller Esercizi con i microcontroller

Cavi e accessori

Materiali allegati a questo fascicolo.

Con questo fascicolo viene fornito un altro cavetto di collegamento, terminato su una coppia di connettori a quattro terminali, oltre a del filo rigido per aumentare le possibilità di collegamento tra i terminali della Bread Board e da questa verso l'esterno. A questo si aggiungono i due connettori e i due condensatori di filtro dell'alimentazione per completare la scheda del driver DG02.

Togliamo le viti delle schede DG01 e DG02, e allentiamo quelle di DG04.

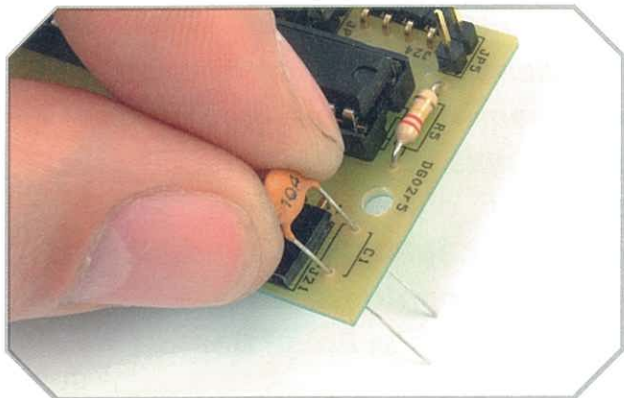
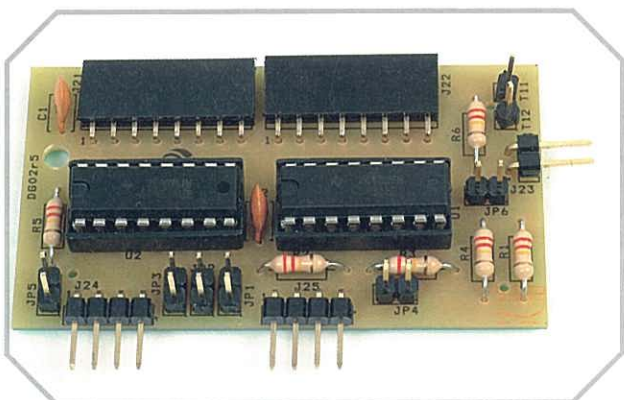
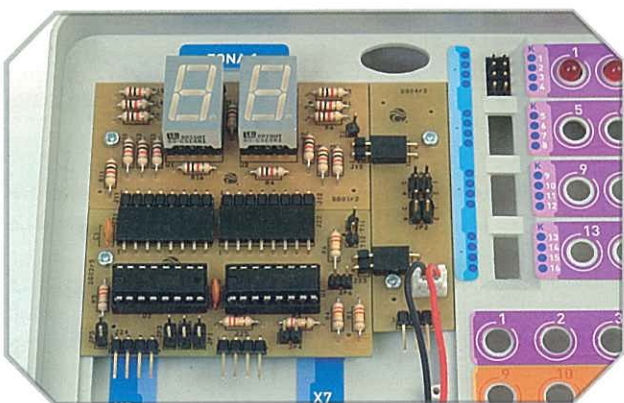
Scheda DG02

Il primo lavoro a cui ci dedicheremo sarà completare il circuito stampato DG02, smontandolo faremo attenzione a scollegare simultaneamente DG01 e DG02 da DG04. È necessario togliere completamente le viti che fissano le schede DG01 e DG02 e allentare di un giro le due viti che fissano la scheda DG04.

Dopo aver tolto la scheda si inseriscono i terminali nei fori indicati sulla serigrafia della scheda come T11 e T12, si volta la scheda e si salda dal lato apposito per la saldatura.

I terminali dei due condensatori di filtro dell'alimentazione, che sono del tipo ceramico da 100 nF di capacità, si inseriscono nei fori indicati sulla scheda come C1 e C2, in seguito si salderanno e si taglierà la parte in eccesso dei reofori di entrambi.

Con queste operazioni la scheda DG02 è totalmente terminata. Questa scheda si assembla con la DG01 ed entrambe si collegano alla DG04. Si montano le viti di fissaggio delle schede DG01 e DG02, dopodiché si stringono le viti della DG04. Queste viti devono essere chiuse stringendole quanto basta per fissare le schede, non devono essere strette più del necessario in quanto si potrebbero danneggiare i filetti che le viti stesse creano nelle torrette di fissaggio delle schede.

La scheda DG02 e i componenti in attesa di essere installati sulla scheda stessa.

C1 e C2 sono i condensatori di filtro dell'alimentazione.

Scheda DG02 completa.

Reinstallazione dei circuiti stampati DG01, DG02 e DG04.

Collegamenti con la Bread Board

Vi è stato fornito un filo verde lungo circa 1 m, lo taglieremo in parti più corte per adattarlo ai collegamenti tra i vari punti della Bread Board e tra questa e gli altri elementi del laboratorio, in special modo con le molle di collegamento che vi verranno fornite prossimamente. Non c'è una regola fissa per fare questi pezzi, dato che sulla Bread Board si può lavorare in molti modi.

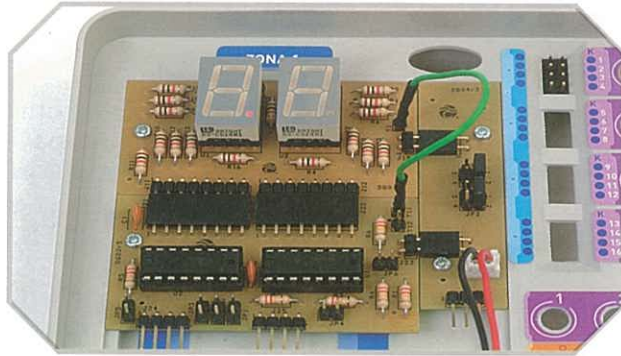
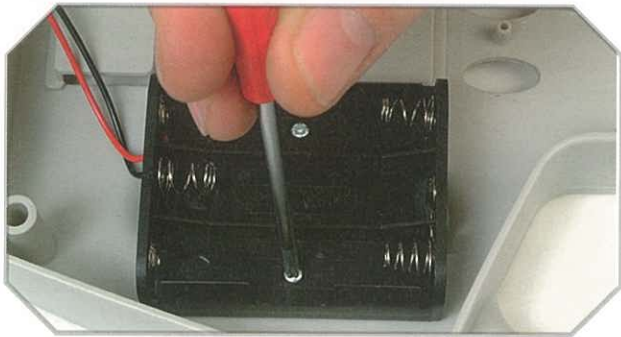
Il nostro consiglio è di dividere questo filo e quello fornito in precedenza per ottenere un insieme di fili di collegamento, spelati agli estremi per circa 6 mm, con le lunghezze indicate di seguito:

12 da 5 cm.
2 da 10 cm.
8 da 15 cm.

Queste misure sono una proposta, non è necessario che siano molto precise, si possono tagliare i fili anche in altro modo, l'importante è che sia possibile realizzare i collegamenti poposti; potete inoltre utilizzare qualsiasi altro filo che non superi 0,5 mm di diametro. Bisogna fare attenzione a non danneggiare il conduttore di rame togliendo l'isolante dalle estremità, per fare in modo che non si rompa.

Portabatterie

Con il fascicolo precedente vi sono state fornite quattro viti, due delle quali sono già state utilizzate, le altre due le utilizzeremo ora per montare il secondo portabatterie.

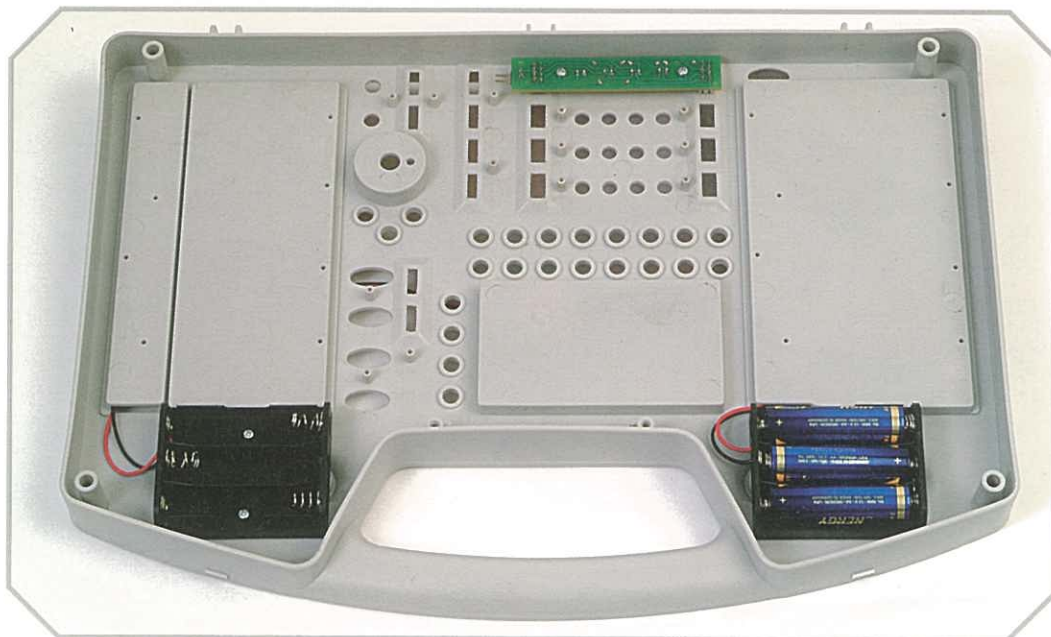
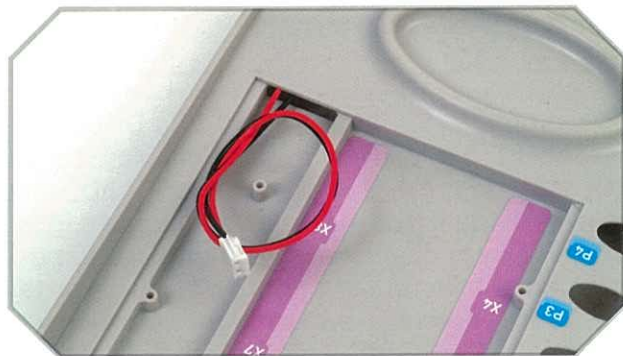
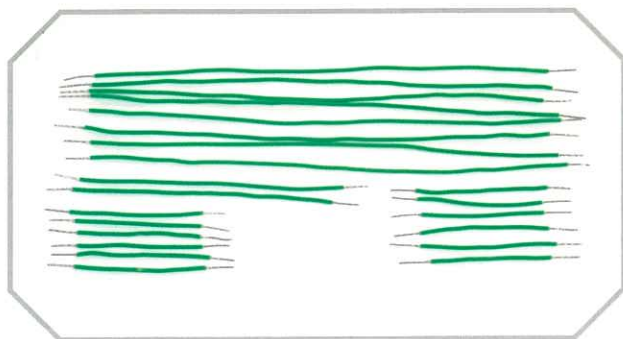
Per installare questo portabatterie gireremo il laboratorio e lo presenteremo nella sua sede, facendo in modo che i suoi fori rimangano allineati con quelli delle torrette del laboratorio, quindi inseriremo una vite e la avviteremo fino in fondo senza stringere, per poter centrare anche l'altro foro; eseguita questa operazione, inseriremo anche l'altra vite e termineremo il fissaggio del portabatterie chiudendo a fondo entrambe le viti. In ogni caso ricordate di non stringere in eccesso.

Prova col filo con due terminali femmina tra il terminale di uscita dell'alimentazione della scheda T1 e uno di quelli appena installati T11 e T12. Il punto di uno dei display si deve accendere.

Il secondo portabatterie si installa utilizzando due viti.

Prova dei terminali T11 e T12

I terminali T11 e T12 della scheda DG02 corrispondono al collegamento della resistenza di polarizzazione dell'anodo del LED, che rappresenta il punto situato nella parte inferiore di ogni display a sette segmenti. Se osserviamo il circuito stampato potremo contare 16 resistenze, una per ogni segmento e due per i due punti. Il driver non è progettato per controllare l'accensione di questi due LED, quindi sarà necessario utilizzare un collegamento esterno quando vorremo farli illuminare.

L'illuminazione di questi punti si ottiene applicando una tensione positiva ai terminali T11 e T12 della scheda DG02. Questa tensione positiva deve essere compresa tra 4,5 e 9 volt e si può ottenere dall'uscita ausiliaria T1 della scheda DG01.

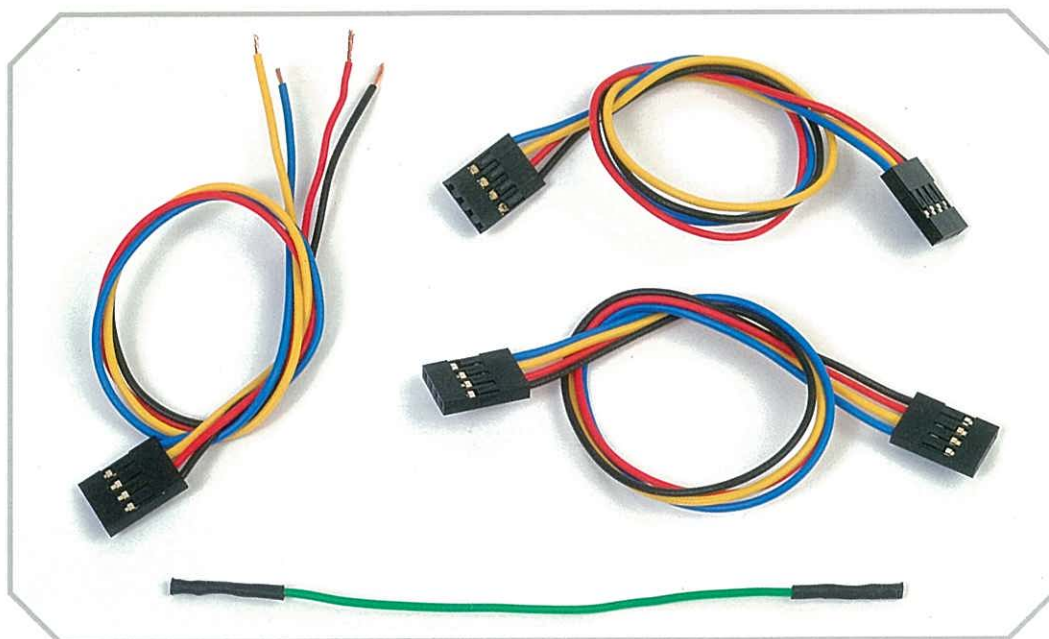
Per realizzare questa prova bisogna montare tre pile da 1,5 volt nel portabatterie situato sotto la zona 1, inoltre bisogna inserire i ponticelli tra 1-2 e 1-2 della scheda DG04.

Se colleghiamo un estremo del filo con connettore femmina sul terminale T1 e l'altro estremo a T11, si illumina il punto di uno dei display, se ci spostiamo su T12 si illumina il punto dell'altro display.

Aspetto del pannello principale del laboratorio visto da sotto.

Per il momento il cavo del portabatterie si deve arrotolare nella zona 2 del laboratorio.

Insieme di fili di connessione.

I cavetti

I cavetti di collegamento con connettori a quattro terminali hanno diverse applicazioni, dato che i collegamenti dei circuiti stampati sono raggruppati, per quanto possibile, di 4 in 4. Anche la scheda dei pulsanti, che non vi è ancora stata fornita, e i collegamenti dei LED, son raggruppati di 4 in 4.

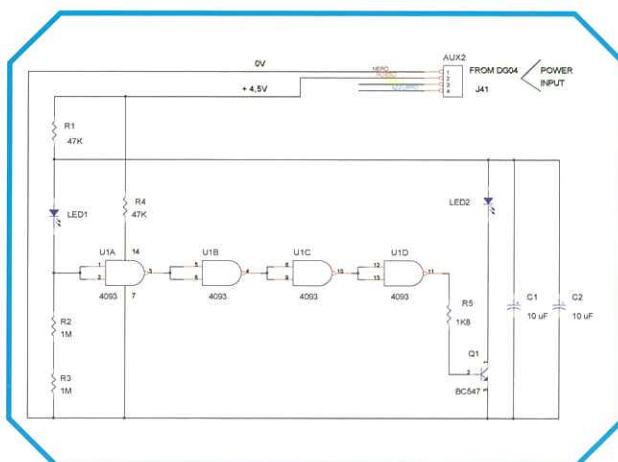
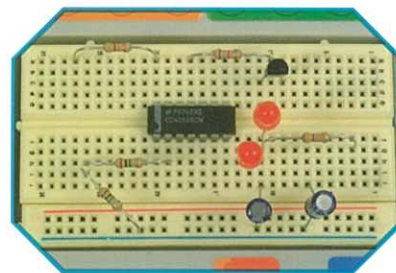
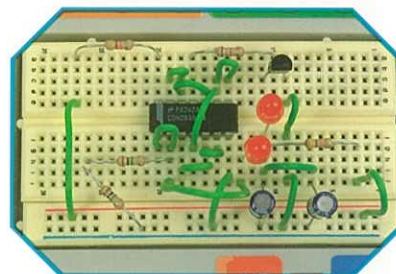
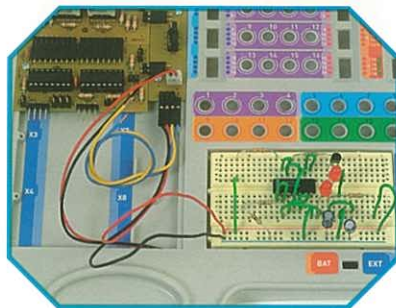
Per quanto riguarda l'alimentazione, li utilizzeremo provvisoriamente in questi fascicoli, dato che collegando tutte le schede l'alimentazione si trasferirà da una all'altra, inoltre le molle indicate come 0 V, 5 V, 9 V e V saranno alimentate dalla parte inferiore del pannello principale, quindi i collegamenti tra la Bread Board e queste molle di connessione si potranno realizzare direttamente con uno o due pezzi di filo con gli estremi spelati. Fino a questo momento vi abbiamo fornito solamente due cavi terminati su due connettori a quattro terminali, uno terminato su un connettore e un cavetto ausiliario che ha un solo filo terminato su un connettore femmina a ogni estremo.

Ve ne forniremo altri con la lunghezza e il numero di terminali necessari per realizzare tutti gli esperimenti proposti, dato che, andando avanti, questi si complicheranno sempre di più, avremo a disposizione più materiali e vi verranno spiegati più argomenti.

Questi sono i cavetti di collegamento già forniti.

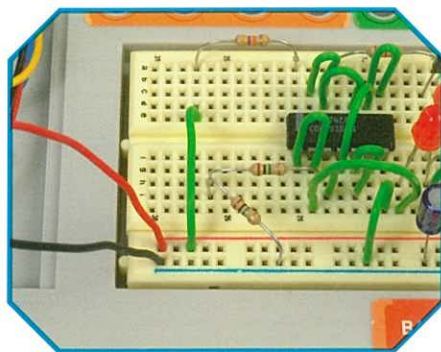
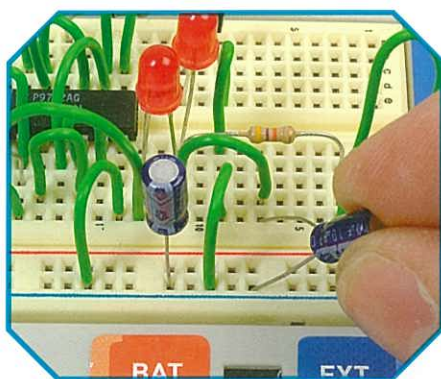
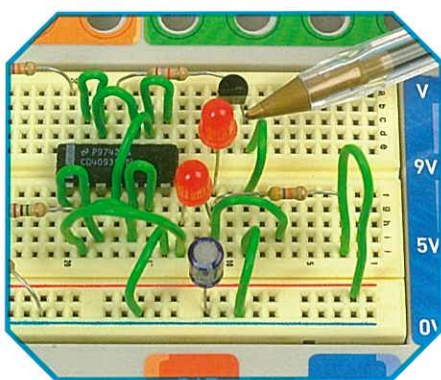
LED a basso consumo

Questo circuito è un esempio di come, complicando un po' l'elettronica, è possibile ottenere un indicatore luminoso a LED in più a basso consumo. Un LED costantemente acceso ha bisogno di almeno 3 mA per potersi illuminare, consumo molto elevato per un dispositivo alimentato a batterie. Con questo circuito invece si dispone di un LED che emette un lampeggio all'incirca ogni 3 o 5 secondi, con un consumo inferiore a 40 mA, quindi può funzionare molti giorni senza cambiare le batterie.

Lo schema

Il circuito è semplice, però un po' particolare. Si tratta fondamentalmente di un circuito di controllo che funziona solamente quando il condensatore C1 + C2 si carica tramite la resistenza R1 da 47 K e inoltre quando all'ingresso del circuito — terminali 1 e 2 dell'integrato — si raggiunge il livello logico 1. Quando questo ingresso passa a livello 1 l'uscita del circuito, terminale 11 del circuito integrato, passa anch'essa a livello 1, portando in conduzione il transistor Q1, quindi il LED 2 si illumina. In questo momento viene consumata in modo quasi istantaneo l'energia accumulata nel condensatore formato dalla somma di C1 e C2. Nello stesso istante, come conseguenza del fatto che il condensatore si è scaricato, la tensione sull'ingresso di controllo si abbassa: il circuito interpreta questo come un livello basso, quindi il LED cessa di condurre. Il LED 1 si utilizza per mantenere una differenza di potenziale e il suo consumo è così ridotto che non si illumina nemmeno.

Schema elettrico del LED a basso consumo.

Componenti inseriti sulla Bread Board.

Cablaggio dei collegamenti.

Alimentazione tramite J41 di DG04, utilizzando questa volta il cavetto di un connettore.

L'attivazione

Quando si collega l'alimentazione del circuito, gli ingressi della porta U1A — terminali 1 e 2 del circuito integrato — sono a livello basso, e le resistenze R2 e R3 li mantengono a livello basso. Questa situazione dura piuttosto a lungo, perché la carica del condensatore è lenta, in quanto si produce tramite una resistenza da 47 K. Tramite il LED 1 non circola corrente, quindi non c'è attivazione fino a quando non viene superata la sua soglia di conduzione, do-

Dettaglio del collegamento della alimentazione.

Se si toglie un condensatore aumenta la frequenza del lampeggio.

Lampeggia solamente il LED 2.

Laboratorio con l'esperimento montato.

podiché la corrente che circola è minima, sufficiente per l'attivazione del circuito che origina il lampeggio, ma insufficiente a fare illuminare questo LED. Le porte dell'integrato si utilizzano come invertitori, dato che hanno i loro due ingressi uniti fra loro, quindi invertendo un segnale un numero pari di volte, il segnale rimarrà com'era; ma con un certo ritardo che si accumula di porta in porta. Utilizzando una porta CMOS, il consumo è molto basso, dato che questo tipo di porte ha un'impedenza d'ingresso molto alta, invece la loro uscita può fornire corrente sufficiente a pilotare la base di un transistor e portarlo in conduzione.

Montaggio

Il montaggio è simile agli altri già realizzati, anche se il cablaggio si complica maggiormente e bisogna avere maggior attenzione alla polarità dei LED, a quella dei condensatori elettrolitici e al posizionamento del transistor.

Alimentazione

Il circuito ha un consumo molto ridotto, che si avvicina a malapena a 40 µA. Il circuito integrato si alimenta tramite una resistenza da 47 K, questo è possibile grazie al suo basso consumo. Il diodo LED 2 non ha resistenza di limitazione, in quanto l'energia che deve dissipare è limitata dall'energia accumulata nel condensatore. Se si desidera un lampeggio più luminoso è sufficiente aumentare la capacità del condensatore C1, o diminuire il valore di R1.

Modifiche

Se riduciamo il valore del condensatore, togliendone uno e lasciando solamente l'altro da 10 µF collegato in parallelo, diminuisce l'energia del lampeggio, anche se a prima vista è difficile notare questo cambiamento, però la carica sarà più veloce, quindi aumenterà la frequenza del lampeggio stesso.

LISTA DEI COMPONENTI

Circuito di base	
U1	Circuito integrato 4093
R1, R4	Resistenza 47 K (giallo, viola, arancio)
R2, R3	Resistenza 1 M (marrone, nero, verde)
R5	Resistenza 1K8 (marrone, grigio, rosso)
C1, C2	Condensatore 10 µF elettrolitico
Q1	Transistor NPN BC547 o BC548
LED1, LED2	Diodo LED rosso

Simulazione con MPLAB (II): finestre

Continuiamo l'analisi delle possibilità che ci offre MPLAB quando si simula il funzionamento di un programma. Abbiamo visto quali sono le finestre di simulazione, ma ci rimangono ancora da studiare alcune di esse che risultano fondamentali in questo processo.

File Registers

Quando nella barra dei menù selezioniamo Window, appare un menù a tendina dove di seguito alle cose già viste, apparirà l'opzione File Registers. In questa finestra MPLAB presenta una lista di tutti i registri di utilizzo generale (GPR) del microcontroller. Come succedeva con alcune delle finestre viste in precedenza, ci sono tre modi di visualizzazione che si possono selezionare con l'icona situata nell'angolo superiore sinistro.

Hex Display: questa opzione visualizza i registri con i dati in formato esadecimale, così come li possiamo vedere nella figura riprodotta qui sotto.

Symbolic Display: questo formato presenta i registri di utilizzo generale con le loro etichette, se le hanno, e il loro contenuto in esadecimale, decimale, binario e formato carattere.

È la visualizzazione più pratica perché oltre alla posizione nella memoria possiamo identificare facilmente sia il registro che il suo contenuto.

ASCII Display: in questo caso i dati dei registri sono presentati in codice ASCII.

Tramite questa finestra potremo modificare il contenuto di uno o più registri. Se puntiamo con il mouse sul primo registro che vogliamo modificare e lo selezioniamo mediante il pulsante destro, si attiverà l'opzione Fill Register(s). Questa opzione ci permetterà di modificare il valore del registro; se con il mouse selezioniamo tutto un blocco e clicchiamo il pulsante destro possiamo selezionare diversi registri.

Più avanti commenteremo questa comoda opzione.

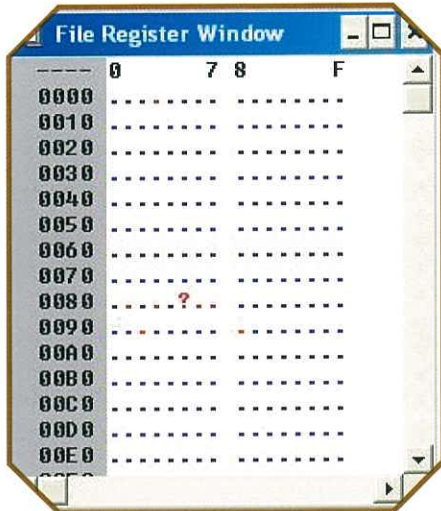
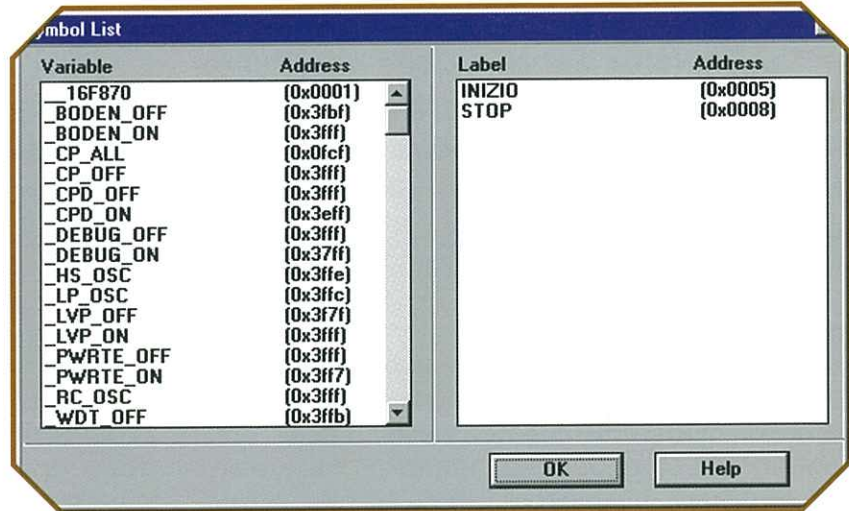
File Register Window

----	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
0000	00	00	00	18	00	00	00	00	--	--	00	00	00	00	00	00
0010	00	00	00	--	--	00	00	00	00	00	00	--	--	--	00	00
0020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0080	00	FF	00	18	00	3F	FF	FF	--	--	00	00	00	00	00	--
0090	--	--	FF	--	--	--	--	--	02	00	--	--	--	--	00	00
00A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00C0	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
00D0	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
00E0	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
00F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0100	00	00	00	18	00	--	00	--	--	--	00	00	00	00	00	00
0110	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
0120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0180	00	FF	00	18	00	--	FF	--	--	--	00	00	00	00	--	--
0190	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
01A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
01B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
01C0	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
01D0	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
01E0	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
01F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Listato dei registri di utilizzo generale del sistema.

ile Register Window

0000	HEX	DEC	BINARY	CHAR	SYMBOL NAME
0001	00	0	00000000	.	__16F870
0002	00	0	00000000	.	PS2
0003	18	24	00011000	.	NOT_PD
0004	00	0	00000000	.	NOT_TO
0005	00	0	00000000	.	TXEN
0006	00	0	00000000	.	ADCS0
0007	00	0	00000000	.	ADFM
0008	--	---	--------	-	
0009	--	---	--------	-	
000A	00	0	00000000	.	PCLATH
000B	00	0	00000000	.	INTCON
000C	00	0	00000000	.	PIR1
000D	00	0	00000000	.	PIR2
000E	00	0	00000000	.	TMR1L
000F	00	0	00000000	.	TMR1H
0010	00	0	00000000	.	T1CON
0011	00	0	00000000	.	TMR2
0012	00	0	00000000	.	T2CON
0013	--	---	--------	-	
0014	--	---	--------	-	
0015	00	0	00000000	.	CCPR1L
0016	00	0	00000000	.	CCPR1H
0017	00	0	00000000	.	CCP1CON
0018	00	0	00000000	.	RCSTA
0019	00	0	00000000	.	TXREG
001A	00	0	00000000	.	RCREG
001B	--	---	--------	-	
001C	--	---	--------	-	
001D	--	---	--------	-	
001E	00	0	00000000	.	ADRESH
001F	00	0	00000000	.	ADCON0
0020	00	0	00000000	.	RISULTATO
0021	00	0	00000000	.	
0022	00	0	00000000	.	
0023	00	0	00000000	.	
24	00	0	00000000	.	

Registri di utilizzo generale nel modo visualizzazione Symbolic.

I registri di utilizzo generale in modo ASCII.

Listato dei simboli utilizzati nel programma.

Special Function Registers

Selezionando Window → Special Function Registers potremo vedere il contenuto dei registri con funzioni speciali (FSR). Durante la simulazione questa è una delle finestre da tenere normalmente aperta, il formato fornito da questa finestra risulta molto utile per analizzare il contenuto di questi registri in ogni momento. Come mostra la figura, in questa finestra appaiono tutti i registri FSRs identificati con il loro nome e con il loro contenuto nei formati esadecimale, decimale, binario e ASCII.

Per modificare il contenuto di uno di questi registri dobbiamo fare doppio clic sul registro prescelto e immediatamente apparirà la finestra Modify, che permetterà la scrittura del registro.

Special Function Register Window

SFR Name	Hex	Dec	Binary	Char
w	00	0	00000000	.
tmr0	00	0	00000000	.
option_reg	FF	255	11111111	.
pcl	00	0	00000000	.
pclath	00	0	00000000	.
status	18	24	00011000	.
fsr	00	0	00000000	.
porta	00	0	00000000	.
trisa	3F	63	00111111	?
portb	00	0	00000000	.
trisb	FF	255	11111111	.
portc	00	0	00000000	.
trisc	FF	255	11111111	.
intcon	00	0	00000000	.
pir1	00	0	00000000	.
pie1	00	0	00000000	.
pir2	00	0	00000000	.
pie2	00	0	00000000	.
tmr1l	00	0	00000000	.
tmr1h	00	0	00000000	.
t1con	00	0	00000000	.
tmr2	00	0	00000000	.
pr2	FF	255	11111111	.
t2con	00	0	00000000	.
ccpr1l	00	0	00000000	.
ccpr1h	00	0	00000000	.
ccp1con	00	0	00000000	.
rcsta	00	0	00000000	.
txreg	00	0	00000000	.
rcreg	00	0	00000000	.
txsta	02	2	00000010	.
spbrg	00	0	00000000	.
resh	00	0	00000000	.

Finestra dei registri di funzione speciali (FSRs).

Show Symbol List

In questa finestra sono riportati i simboli delle variabili e delle etichette, utilizzati nel codice sorgente del nostro programma. Dopo aver compilato il programma si genera un file con estensione ".cod" che contiene la definizione di tutti i simboli utilizzati nel programma. Questa finestra ha una funzione informativa riguardo ai simboli che abbiamo utilizzato e serve unicamente come orientamento in modo da poter identificare gli indirizzi di memoria a cui sono associati questi simboli.

Stopwatch

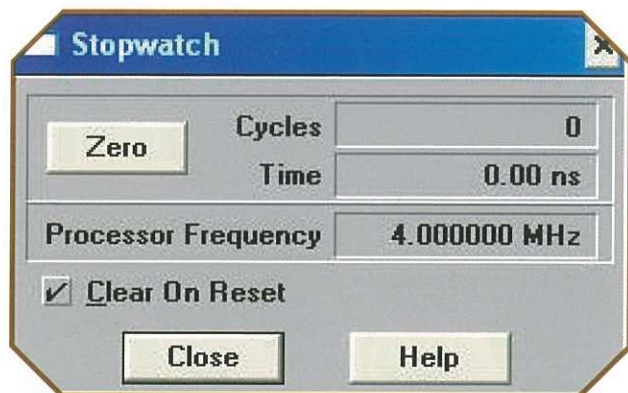
Durante la simulazione spesso ci tornerà utile poter calcolare il tempo di esecuzione di un programma o di una subroutine. Possiamo fare questo contando il numero di istruzioni che si realizzano e moltiplicandolo per quattro volte la frequenza del segnale di clock, o per otto nel caso in cui le istruzioni siano di salto, oppure lo potremo fare utilizzando il nostro software. MPLAB offre la possibilità di conta-

Stopwatch

Zero | Cycles 0 | Time 0.00 ns

Processor Frequency 4.000000 MHz

Clear On Reset

Close | Help

Finestra Stopwatch per contare il tempo di esecuzione delle istruzioni del nostro programma.

Tramite questa finestra è possibile cambiare la configurazione dell'oscillatore del microprocessore.

re il tempo di esecuzione delle istruzioni del programma mediante questo utile "cronometro", che calcola il tempo in base alla frequenza definita per il microcontroller che stiamo simulando. Per iniziare a utilizzare questo software è necessario definire la frequenza di lavoro per il microcontroller e questa impostazione rimarrà memorizzata anche per il resto dei progetti. Nel caso in cui si voglia modificare questa frequenza bisognerà eseguire l'operazione tramite il menù Option → Development Mode e dopo aver aperto questa finestra bisognerà selezionare la scheda Clock. Qui potremo scegliere il tipo di oscillatore da simulare e la frequenza di lavoro.

Possiamo mantenere aperta la finestra Stopwatch durante la simulazione per verificare i cicli e il tempo che impiega a essere eseguita ogni istruzione oppure il programma completo.

Project Window

```
Project Listing
Path:                    C:\PROGRA~1\MPLAB\PROGETTI\
Project Name:            ESER2.PJT
Target:                  ESER2.HEX
Tool Suite:              Microchip
Processor:               PIC16F870
Development Mode:        Simulator

Target Data
File List:               ESER2.ASM;
Build Tool:              MPASM

Node:                    ESER2.ASM
Dependency List:
                         P16F870.INC
```

Finestra di progetto.

Project Window

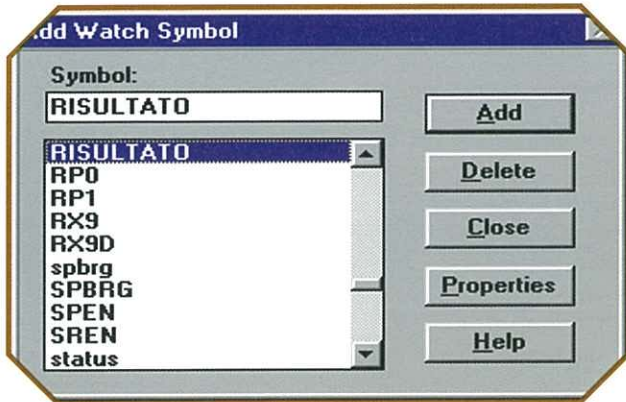
La finestra del progetto serve per presentare i dati di riferimento del progetto che abbiamo aperto. Essa ci indica l'indirizzo dove il progetto è memorizzato, i nomi dei file relativi e parte della configurazione del software per il progetto in questione. Se il progetto è stato assemblato o compilato, in questa finestra sono riportati tutti i file inclusi in esso. Se selezioneremo uno dei file riportati nella finestra, mediante un doppio clic del mouse, potremo aprire questo file per un'eventuale revisione.

Questa finestra ha una funzione informativa ed è raramente utilizzata durante il lavoro con MPLAB.

Watch Window

Selezionando questa opzione sul menù si apre un altro sottomenù dove potremo scegliere cinque opzioni.

New Watch Window: permette di creare una finestra temporanea per visionare i contenuti dei registri che desideriamo. Se durante la simulazione desideriamo vedere il contenuto di alcuni registri particolari, creeremo una finestra di questo tipo e selezioneremo i registri considerati importanti per l'analisi del programma.

Load Watch Window: le finestre create si possono salvare. Quando selezioniamo questa opzione stiamo informando il simulatore che desideriamo caricare una finestra precedentemente configurata, una finestra che abbiamo creato e che si trova memorizzata sull'Hard Disk del computer. Le finestre si memorizzano con estensione ".wat".

Mediante questa opzione possiamo aggiungere i registri che desideriamo analizzare in una finestra.

Aspetto che avrà la nostra finestra dopo aver aggiunto i registri.

Quadro di dialogo per modificare le proprietà della presentazione nella finestra.

Add to Active Watch: questa opzione si abilita solamente se abbiamo già una finestra aperta e permette di aggiungere nuovi registri alla finestra che abbiamo creato.

Edit Active Watch: come la precedente anche questa opzione si attiva solamente quando abbiamo una finestra già aperta e permette di editare i registri che sono stati inseriti nella finestra creata.

Sae Active Wacth: mediante questa opzione possiamo salvare sull'Hard Disk la finestra che abbiamo creato; inoltre possiamo salvare la finestra quando dopo averla creata vogliamo chiuderla. Il software non ci chiederà al momento della chiusura se vogliamo salvare la finestra.

Per aggiungere registri alla finestra dobbiamo selezionare il registro e cliccare sull'opzione Add. Possiamo aggiungere tanti registri quanti ne desideriamo, o creare tante finestre quante ne consideriamo opportune. Normalmente i registri vengono raggruppati per funzioni comuni, all'interno di diverse finestre per la loro analisi.

Possiamo aggiungere dei registri mediante Add, però possiamo anche eliminare un registro che abbiamo aggiunto mediante l'opzione Delete. Inoltre potremo scegliere le proprietà della visualizzazione del registro mediante l'opzione Properties, o selezionando Edit Active Watch e facendo clic su Properties.

Come per la maggior parte delle finestre di simulazione, anche qui troviamo nell'angolo superiore sinistro l'icona del menù di controllo. Mediante questo menù possiamo aggiungere il numero di linee che desideriamo o inserire, cancellare o editare uno dei registri della finestra. Tramite questo menù e possibile inoltre salvare la finestra.

Conclusioni

Termineremo lo studio delle finestre di simulazione analizzando l'opzione che permette di modificare il valore di un registro e vedendo anche le diverse visualizzazioni durante la simulazione. Avendo chiaro le opzioni di analisi che ci offre MPLAB tratteremo le opzioni specifiche della simulazione e infine lavoreremo con esempi in modo da poter sfruttare al massimo questo potente strumento.

Programmazione

Continuiamo con l'introduzione alla programmazione che stiamo alternando alla conoscenza del PIC16F870, affrontando, in questo numero, le strutture di programmazione, i salti condizionali e incondizionali, i procedimenti e le funzioni.

Strutture di programmazione

Le strutture di programmazione, anche chiamate strutture di controllo, sono gli elementi che permettono di eseguire nel programma alcune azioni oppure altre, secondo determinate condizioni. L'utilizzo di queste strutture provoca la non linearità dei programmi che, pertanto, non saranno eseguiti sempre nello stesso modo.

Possibili strutture di controllo per l'esempio di lettura della temperatura.

Diversi tipi di strutture condizionali.

Immaginate un programma che acquisisce la temperatura esterna. In funzione della temperatura attiverà la caldaia, oppure non farà nulla, o attiverà l'aria condizionata. Il modo più logico di realizzare questo programma è attraverso una struttura di controllo che potrà assumere diverse forme, come mostrato nella figura.

Le strutture di controllo si possono classificare in due grandi gruppi:

Strutture condizionali

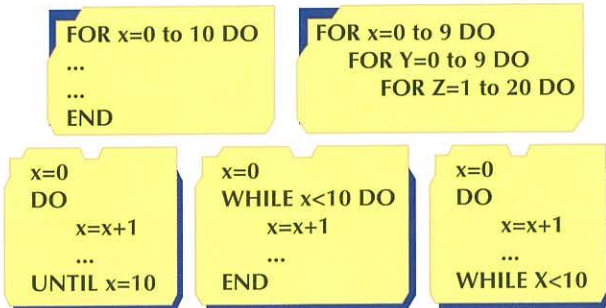
Le strutture condizionali o prese di decisione indirizzano il programma verso un percorso oppure un altro in funzione delle differenti opzioni. Se si compie una serie di condizioni, il programma eseguirà alcune determinate istruzioni.

In base al linguaggio di programmazione utilizzato esisteranno più o meno tipi di strutture. Le più utilizzate sono le "IF...THEN..." e tutte le loro varianti (IF...THEN...ELSE, ELSEIF...) in cui, a seconda se una condizione sia vero o falsa, viene eseguita una determinata azione. Possiamo anche trovare strutture condizionali del tipo "CASE" o "SWITCH", dove le espressioni possono avere valori differenti da vero o falso.

In assembler, le strutture di questo tipo sono i salti condizionali che mantengono la filosofia di IF, dato che rispondono a: "Se si compie questa condizione salta la linea successiva".

Strutture cicliche

Le strutture cicliche o cicli permettono di ripetere lo stesso processo molte volte, senza dover scrivere la stessa linea o linee di codice più volte. Ve ne sono alcune che eseguono un insieme di istruzioni un numero di volte specificato (FOR...DO) e altre che eseguono queste istruzioni fino a o mentre si compie una condizione (WHILE...DO, DO...UNTIL).

È possibile annidare i cicli (mettere un ciclo

Esempi di strutture cicliche.

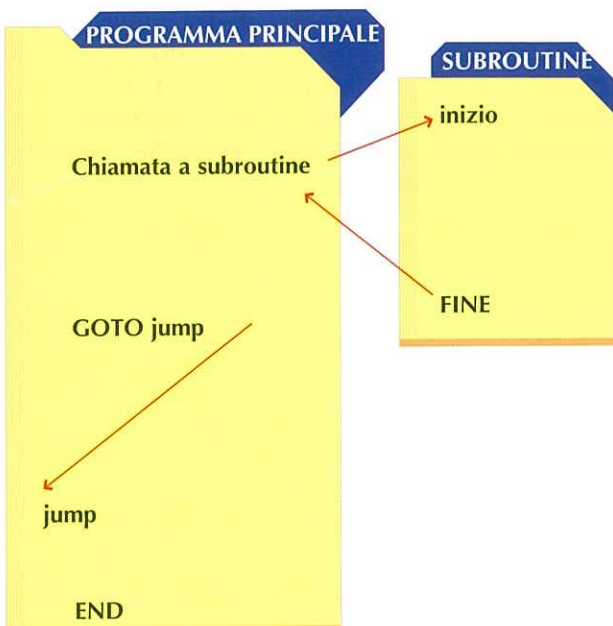
SALTO CONDIZIONALE	DECFSZ registro, spostamento	Decrementa il registro e salta se è 0
	INCFSZ registro, spostamento	Incrementa il registro e salta se è 0
	BTFSC bit, spostamento	Testa il bit e salta se è 0
	BTFSS bit, spostamento	Testa il bit e salta se è 1
SALTO INCONDIZIONALE	GOTO etichetta	Salta all'etichetta

Istruzioni utilizzate in assembler per i salti.

all'interno di un altro) per moltiplicare le interazioni. A seconda del linguaggio di programmazione, avremo a disposizione più o meno istruzioni cicliche.

Nel caso non ci fossero, come succede nell'assembler, le potremo costruire a partire da strutture condizionali e l'istruzione "GOTO".

Salti condizionali e incondizionali

Un salto condizionale è quello che si verifica unicamente se, e solo se, si compie una determinata condizione. Nel linguaggio assembler sono quattro le istruzioni che realizzano un salto condizionale.

Differenze tra chiamata a una subroutine e l'istruzione GOTO.

I salti incondizionali non rispondono a nessuna condizione, si verificano sempre quando appare l'istruzione corrispondente al salto. Nel linguaggio assembler esiste l'istruzione GOTO che genera un salto incondizionale.

Una chiamata a subroutine si può considerare un salto incondizionale, dato che, se si genera, il programma abbandona l'esecuzione del programma principale e salta all'esecuzione della subroutine. La differenza tra l'istruzione GOTO e una chiamata a una subroutine sta nel fatto che, dopo aver eseguito la subroutine, si ritorna al programma principale, mentre con GOTO no.

Quanto più alto è il livello del linguaggio di programmazione, maggiore è la varietà di istruzioni di salto a disposizione.

Procedimenti

I procedimenti sono frammenti di codice che realizzano un determinato compito e che sono dichiarati una sola volta, infatti ogni volta che si vorranno utilizzare sarà sufficiente indicare il loro nome.

Per dichiarare un procedimento, nei linguaggi di programmazione che lo permettono, di solito si utilizza la parola riservata PROCEDURE, seguita dal nome che desideriamo dare al procedimento.

Se lo consideriamo opportuno, possiamo trasmettere variabili a un procedimento tramite il programma principale. A questo scopo dovremo solamente definire, nella dichiarazione di procedimento, una lista con le variabili che desideriamo trasferire. Questo si potrà fare per valore o per riferimento, nel primo caso le variazioni che si producono al termine del procedimento si perderanno; nel secondo

```
programma esempio;
var
base:integer;

procedure     somma(uno, due:integer);

              var
              sol:integer;

              begin
              sol=uno+due;
              writeln ('La somma è :',sol);
              end;

begin
clrscr;
base :=1;
somma(14,12);
somma(25,base);
end.
```

Esempio di programma con chiamata a procedimento in un linguaggio di alto livello.

```
Dichiarazione variabili
  INTERO: N, somma
  fine dichiarazione variabili

inizio

da N=1 fino a N=200 fare
  Somma <- Somma_N_Naturali(N);
  mostra sul display ('La somma dei ',N,'
  primi naturali è ','Somma)
  fine da

fine

funzione Somma_N_Naturali(INTERO N) :
INTERO
  variabili
  INTERO: Somma,i

  Somma <- 0
  da i<-1 fino a i=N fare
    Somma <- Somma+i

  Risultato <- Somma
fine funzione
```

Dichiarazione di una funzione.

```
funzione NOME (arg1,...,argN) : TIPO
  variabili
  ......... {si dichiarano}

    azione 1
    .........
    azione N
  Risultato <- Valore
Fine funzione
```

Esempio di programma che utilizza una funzione.

si trasmette un indirizzo e, nel caso si verifichino delle variazioni nel contenuto dello stesso, queste saranno permanenti, anche una volta terminata l'esecuzione del procedimento.

Nell'esempio della figura vediamo che nella dichiarazione del procedimento si definiscono due variabili: uno e due, variabili intere. Il procedimento ha la struttura di un programma normale e realizza una o più funzioni (nel nostro caso somma due numeri e presenta il risultato sul display). Nel programma principale si può ripetere la chiamata al procedimento scrivendo il suo nome e trasferendo i parametri con cui desideriamo che lavori.

Funzioni

Le funzioni sono molto simili ai procedimenti, con la differenza che restituiscono un valore al termine dell'esecuzione. Il tipo di questo valore si dichiara al momento di dichiarare la funzione. Quindi, una funzione si definirà nel modo indicato dalla figura.

TIPO è il tipo di dato che restituirà la funzione, NOME è il nome dato alla funzione e arg1...argN sono i parametri che passeremo alla funzione. Al termine di tutte le azioni la funzione restituirà un risultato che sarà definito quando chiameremo la funzione.

Per chiamare una funzione dovremo aver definito nelle nostre dichiarazioni di variabili una variabile dello STESSO tipo di quello restituito dalla funzione. Quindi, dovremo assegnare a questa variabile ciò che ci restituisce la funzione nel seguente modo: Variabile <- Nome_Funzione (arg1,...,argN).

La struttura commentata, che si può vedere nell'esempio della figura, è quella corrispondente ai linguaggi di alto livello, e si semplifi-

```
          org 0x00
          goto Inizio
          org 0x05

tabella:  addwf   PCL,F
          retlw   b'00111111'    ;Dig 0
          retlw   b'00000110'    ;Dig 1
          retlw   b'01011011'    ;Dig 2
          retlw   b'01001111'    ;Dig 3
          retlw   b'01100110'    ;Dig 4
          retlw   b'01101101'    ;Dig 5

Delay20:  bcf     INTCON.T0IF    ;Resetta il flag
                                  di overflow
          movlw   0xb1           ;Complemento hex.
                                  di 78
          movwf   TMR0           ;carica il TMR0
Del20_1:  clrwdt                 ;Aggiorna il WDT
          btfss   INTCON,T0IF    ;Overflow del
                                  TMR0 ??
          goto    Delay_20_ms_1;Ancora no
          return

Inizio:   ...
          ...
          ...
          call    tabella
          ...
          ...
          call    Delay20
          ...
          ...
          end
```

Esempio in assembler di utilizzo di funzioni.

Si ottiene una maggior chiarezza del codice.
Si evita la ripetizione inutile di frammenti uguali di codice.
Si dividono le azioni complesse in sottoazioni più semplici.
Si ottiene una maggior modularità e quindi il programma si modifica più facilmente.
Si risparmia memoria utilizzando variabili locali invece di globali.

Vantaggi dell'utilizzo di procedimenti e funzioni.

ca molto quando utilizziamo l'assembler come linguaggio di programmazione. Nella figura allegata possiamo verificare come si dichiara una funzione e si realizza una chiamata in linguaggio assembler.

Recursività e concatenazione

La recursività è la proprietà di una funzione o di un subprogramma di eseguire una chiamata a se stessa. Quindi, all'interno di una funzione, potremo creare una chiamata alla stessa funzione, dato che questa conserverà il valore dell'esecuzione precedente.

Le funzioni o subprogrammi, così come le strutture di controllo, possono essere concatenate, in modo che una funzione sia inclusa all'interno di un'altra, quindi sarà visibile solamente per quest'ultima.

Generalità su procedimenti e funzioni

Molte volte si confondono questi termini e si utilizzano indistintamente per indicare una parte del codice che è fuori, o separata, dal programma principale.

Abbiamo imparato a differenziarle, e le uniche cose che hanno ancora in comune sono i vantaggi offerti dal loro utilizzo.

Ad esempio si risparmia memoria, perché una variabile che si definisce all'interno del procedimento o funzione (locale) avrà senso solamente all'interno di questa e sparirà al termine dell'esecuzione.

Conclusioni

Abbiamo terminato il ripasso della programmazione generale studiando le strutture di controllo, condizionali o cicliche, i salti condizionali e incondizionali, i procedimenti e le funzioni. Questi concetti vi saranno utili per qualsiasi linguaggio di programmazione che utilizzerete e costituiscono la base di un buon programmatore. Analizzeremo e studieremo il repertorio di istruzioni del nostro microcontroller, in modo che, una volta conosciute le 35 istruzioni che possiede, le caratteristiche e i dispositivi del PIC, potremo iniziare a sviluppare applicazioni con esso.